Journal of
open research software

Abhisek, H 2014 Image Enhancer: A Graphic Editor to Apply Numerous Effects in Digital Image.
*Journal of Open Research Software,* 2: e31, DOI: http://dx.doi.org/10.5334/jors.bm

SOFTWARE METAPAPER

# Image Enhancer: A Graphic Editor to Apply Numerous Effects in Digital Image

Hazra Abhisek[1]

[1] Centre for Development of Advanced Computing, India

Image Enhancer is an open source, portable graphic editor developed for Windows platform. It is equipped with an enriched set of digital imaging filters with advanced computer vision techniques embedded within, like Interest Point Detection (Susan Corner Detector), Linear Edge Detection (Simple, Sobel, Canny), Histogram Equalization, Dithering (Bayer, Burkes, Sierra, Jarvis Judis Ninke), Transforming to Polar images and vice versa etc.

Image Enhancer was released under GNU Lesser General Public License (LGPL) and the software was made available from the Microsoft's open source project hosting repository Codeplex (http://imageenhancer.codeplex.com). Image Enhancer was tested and hosted by several popular software archives like SoftPedia, CNET, Freeware Files, ZDNet, Soft Tango and others. A stable Release Candidate (RC) version has been made available in which some major modifications were done which were not present in the earlier Beta version. The download link for the Image Enhancer (both Release Candidate & Beta Version) from CodePlex repository is (http://imageenhancer.codeplex.com/releases).

**Keywords:** image processing, digital filter, computer vision, graphic editor, photo effect

## (1) Overview

### Introduction

Image Enhancer was developed in keeping certain perspectives in mind. Even the openCV, being one the finest real time computer vision library in recent times, does not come with an "out-of-the-box" GUI, apart from some burning issues like complex usability, memory management schemes and developmental environment dependencies. Since there are a very limited number of open source software for performing image processing research so the need for incorporating all digital imaging filters integrated in one place was one of the major goals of this development. As a result, the researchers in the fields of image processing and computer vision could utilize it in their research or developers may extend the code for specific application development. Also the ability to edit the parameters of an image processing algorithm is another important aspect in Image Processing research which increases the reusability of the algorithm significantly. The parameters of the digital imaging filters can be customized according to end user choices so that the reusability and popularity of the software increases for the researchers/developers who want to develop an image processing workflow. Currently, it also provides support for several image file types apart from jpg and bmp. The source code of Image Enhancer Release Candidate version can be found here (http://imageenhancer.codeplex.com/SourceControl/latest). It was awarded with "100% FREE" & "100% CLEAN" awards by several popular software archives of the world.

Using Image Enhancer is quite simple. The end user navigates through the three interfaces by just pressing the "Next" or "Back" buttons. If any problem occurs during image conversions then the errors and exceptions are handled in a proper manner with relevant warning or error dialogues. Users can also use the "Refresh" button to reload the picture viewer area. As an example, when an input image is loaded in picture viewer area (**Fig. 1**), we choose a specific filter to be applied upon the image. When we press the specific filter button, a separate dialogue appears for us to choose parameter values of that specific filter (**Fig. 2**). When the threshold values are chosen, users can get their desired image as per their requirements (**Fig. 3**). Similarly, the video buffering interface has a drop down list in which the users can select the USB/Surveillance camera according to their choice from a list of cameras, if connected to the computer. 'Start', 'Stop' buttons starts and stops video buffering in the picture box viewer respectively. 'Save' button saves the snapshot to the disk drive after stopping the video buffering. The user interface was built in Windows Form which provides access to native MS Windows interface elements by wrapping the Windows API in managed code as a replacement of old and complex Microsoft Foundation Class (MFC) library in C++. The code was written in Visual C# and

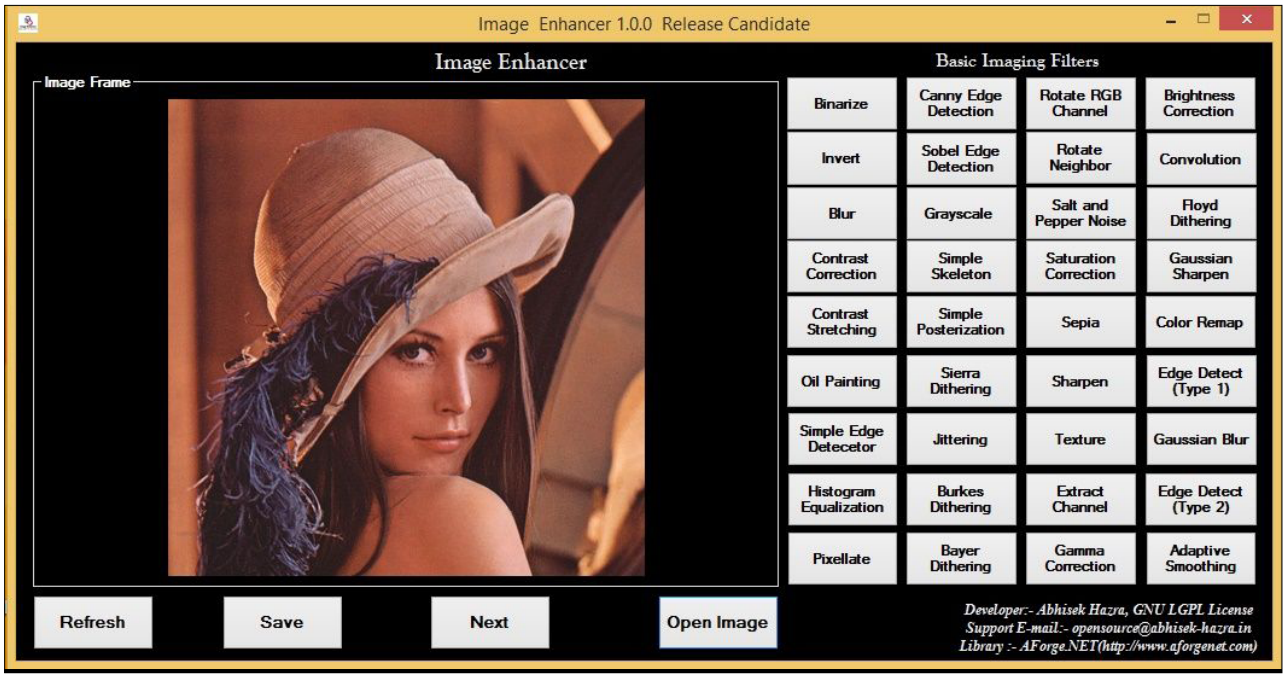


**Fig. 1**: RGB Input Image in Image Enhancer. © Playboy. All Rights Reserved.

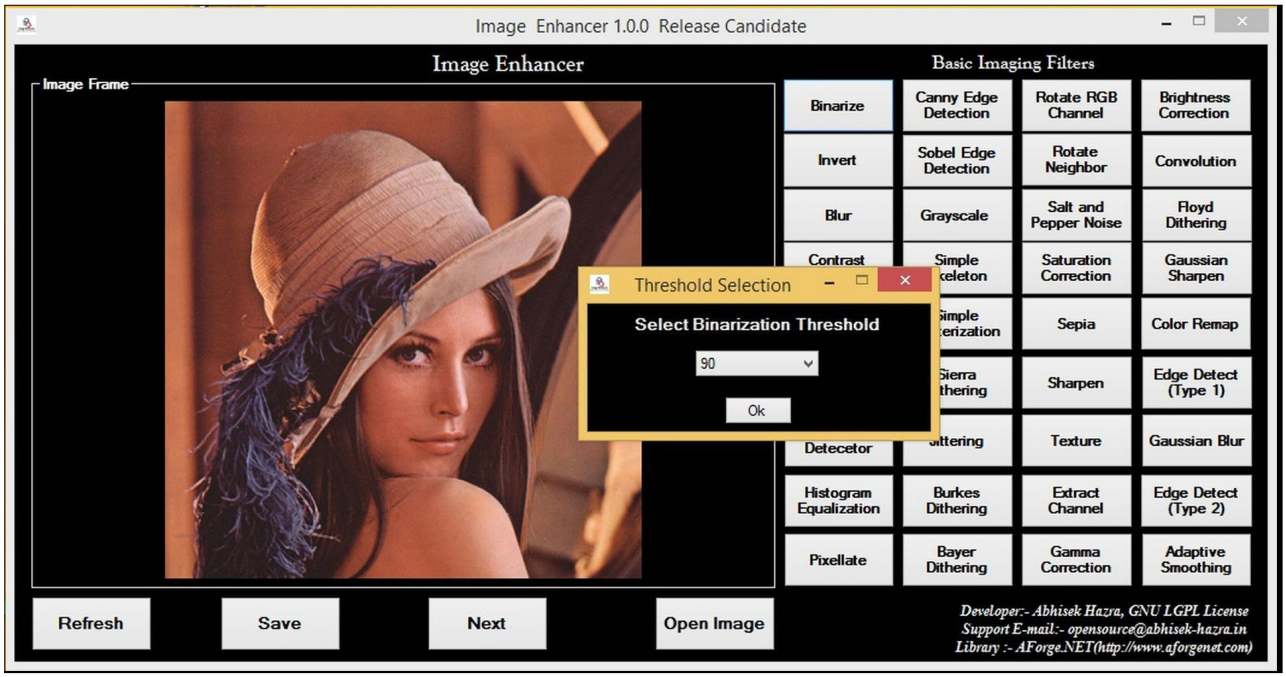


**Fig. 2**: Threshold selection for Binarization. © Playboy. All Rights Reserved.

fine-tuned for optimal time and space complexity to support faster execution of the software.

Image Enhancer contains 64 simple and complex imaging filters covering well known image enhancement techniques (e.g. brightness correction, increment or decrement, contrast correction or stretching, RGB scale rotation), Linear Edge Detection Techniques (Simple, Canny, Sobel), Dithering (Simple, Burkes, Sierra, Stucki, Jarvis Judis Ninke), Interest Point Detection (Susan Corner Detector) and many more. Almost all the parameters of an image processing algorithm can be modified generically as per end user requirements. Additionally users can use the video buffering from USB/Surveillance camera to take snapshots from video and apply effects to that image. Currently Image Enhancer 1.0 RC accepts *.jpg, .bmp, .png, .tiff, .tif, .emf, .exif, .gif, .wmf* and *.dcm* image formats. More imaging formats to be added later in the upcoming version.

The GUI was developed in Windows Forms Application as a part of Microsoft .NET framework. This software was written using Visual C# 4.0 using Visual Studio 2010 IDE. AForge.NET (http://www.aforgenet.com), an open source

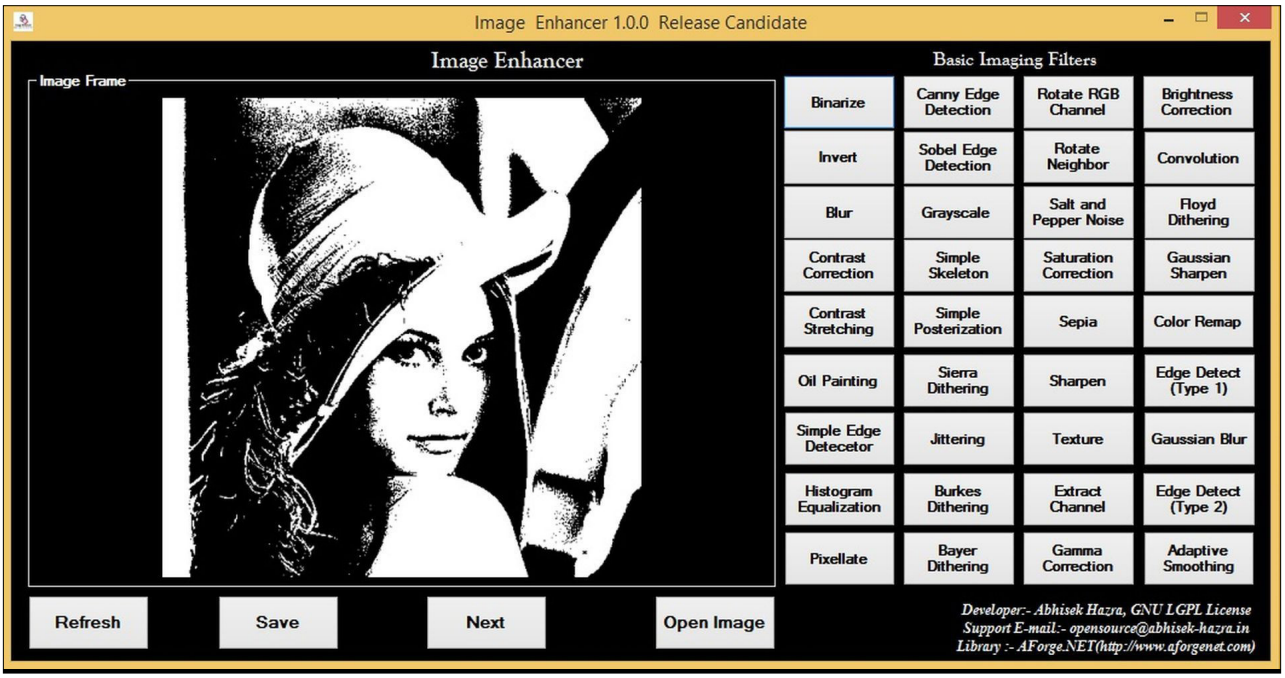


**Fig. 3**: Binarized Output Image with a threshold limit of 90. © Playboy. All Rights Reserved.

computer vision library was used to develop this software (**Ref. 3**).

Image Enhancer (Both RC and Beta version) has a stable release from Microsoft CodePlex code repository. It runs in Windows Platform (Windows XP Service Pack 3, Windows Vista, and Windows 7/8/8.1). Microsoft Dot NET client profile is the only dependency which is required to run this software.

## Implementation and architecture

Image Enhancer was written using C# and Microsoft .Net 4.0 framework which allows high performance, scalable and robust software development for windows platform. The main files are Form1.cs, Form2.cs & Form3.cs where the events for filter invocation of all the filters were written. Each filter was implemented in its own separate class e.g. Image Binarization was implemented in Binarization.cs, Adaptive Smoothing in AdaptiveSmooth.cs and so on. Form1.cs.design, Form2.cs.design & Form3.cs.design are the designer classes contain the code for the three interfaces. When the software runs, the first GUI is loaded. Using the "Open Image" button, which is basically an "OpenFileDialogue" control in Windows Form, the input image is loaded into the image frame (**Fig. 1**). The GUI is quite simple and user friendly, which contains all the buttons and functionalities for digital imaging filters. The output of the applied filter is shown in the picture box to get a widening idea about the effect of the applied imaging filter. The GUI has the "Save" button option through which the filtered image may be saved to disk drive according to the choices of the users. Also the snapshot from a video source (USB/Surveillance Camera) may be taken and saved using the third form interface.

### *Using the code*

Code section of Image Enhancer is located at Microsoft CodePlex repository (http://imageenhancer.codeplex.com/SourceControl/latest) under the GNU Lesser General Public License (LGPL). In the following code example, Binarization filter is applied on the input image. At the very beginning, all namespaces required for this project in .NET were imported into the workspace. At first, when the "Binarize" button is clicked from the interface, the Binarize_Click event is invoked (**Line 6**).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using AForge.Imaging.Filters;
using AForge.Math.Random;
using AForge.Imaging.Textures;
using System.IO;
using System.Drawing.Imaging;
using AForge.Imaging;
```

Until the filter is applied on the input image, current cursor icon will be a busy cursor indicating that a task is running in the background (**Line 10**).Then the input image (rgb_image) is copied to a Bitmap object for further processing (**Line 12**). An object of Binarization class is created (**Line 15**) which is a partial windows form class capable of accepting Binarization threshold (0-255 for an 8 bit image) from users. Next an instance of FiltersSequence class from AForge.Imaging.Filters namespace is created to

```
1        /* Method stub for Image Binarization */
2        /* The filter accepts 8, 16 bpp grayscale images for processing */
3       /* In the case of 8 bpp images the threshold value is in the [0, 255] range,
4         * but in the case of 16 bpp images the threshold value is in the [0, 65535] range */
5
6        private void Binarize_Click(object sender, EventArgs e)
7        {
8            try
9            {
10              Cursor.Current = Cursors.WaitCursor;
11              int final_threshold_binarization;    // storing the final binarization threshold
12              Bitmap bp = (Bitmap)rgb_image;
13              if (bp != null)
14              {
15                  Binarization bs = new Binarization();
16                  bs.ShowDialog();
17                  final_threshold_binarization = Convert.ToInt32(bs.getValue());
18                  Bitmap temp = bp.Clone() as Bitmap;
19                  FiltersSequence fseq = new FiltersSequence();
20                  fseq.Add(AForge.Imaging.Filters.Grayscale.CommonAlgorithms.BT709);
21                  temp = fseq.Apply(temp);
22   /* As Threshold filter accepts 8/16 bpp grayscale image, so the input grayscale image is
   converted into 8 bpp(0-255 threshold range) grayscale image before binarization */
23                  if (temp.PixelFormat != PixelFormat.Format8bppIndexed)
24                    temp = new Bitmap(temp.Width, temp.Height, PixelFormat.Format8bppIndexed);
25                  Threshold t = new Threshold(final_threshold_binarization);
26                  t.ApplyInPlace(temp);
27                  picImage.Image = new Bitmap(temp);
28                  picImage.Show();
29                  Cursor.Current = Cursors.Default;
30               }
31                  else
32                  MessageBox.Show("Select the Image", "Close",
     MessageBoxButtons.OK, MessageBoxIcon.Warning);
33                  Cursor.Current = Cursors.Default;
34                  this.Refresh();
35           }
36           catch (Exception e1)
37           {
38                  MessageBox.Show("Error in Binarization", "Try Again",
     MessageBoxButtons.OK, MessageBoxIcon.Error);
39                  this.Refresh();
40           }
41       }
```

**Code Segment 1:** Binarization event present in Form1.cs.

add and apply Grayscale and Threshold filters (**Line 19-26**). Belonging to AForge.Imaging.Filters namespace, The FiltersSequence class, represents collection of filters, which need to be applied to an image in sequence. Using this class, user may specify set of filters, which will be applied to source image one by one in the order users define them. The Grayscale filter converts the RGB image to grayscale, Otsu Threshold filter is used for threshold setting in binarized images and to convert from grayscale image to binary image. These filters are combined in FiltersSequence object and applied on the input image and output image is stored. The resultant binarized image is shown in the picture box viewer (**Line 28**). If the input image to this method stub is null then the user is prompted to browse for the input image (**Line 31-34**). Any errors or exceptions occurred during conversion from RGB to binary image, is properly handled through catch block (**Line 36-40**).

### *Parameter customization of imaging filter*

Like other image editors, editing the threshold parameters of a specific image processing algorithm is also an important aspect of Image Enhancer. The following part (**Code Segment 2**) represents the Binarization class through which users can edit the binarization threshold. Whenever, threshold value selected from the Binarization dialogue control, the SetValue() method (**Line 14-17**) present in Binarization.cs, sets the value of the threshold and GetValue() method (**Line 18-22**) returns the threshold value to the Binarize_Click event. Whenever the user chooses the threshold value from Binarization dialogue, it is ensured whether the user enters a valid threshold value within the range (For an 8 bit image, the RGB intensity varies between 0 -255) (**Line 32-37**) otherwise a warning dialogue appears notifying the user to enter valid threshold values (**Line 38-40**).

Whenever the Binarize_Click event occurs, an object of binarization class is created, through which the GetValue()

```
1        using System.ComponentModel;
2        using System.Data;
3        using System.Drawing;
4        using System.Linq;
5        using System.Text;
6        using System.Windows.Forms;
7
8        namespace Accord_Test
9    {
10    public partial class Binarization : Form
11    {
12       /* threshold value for binarization within the range[0-255] for 8 bit image */
13        private string threshold;
14        public void SetValue(string str)
15        {
16            this.threshold = str;
17        }
18        public string getValue()
19        {
20       /* method returns the binarization threshold */
21          return threshold;
22        }
23        public Binarization()
24        {
25            InitializeComponent();
26        }
27        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
28        {
29        }
30        private void OkButton_Click(object sender, EventArgs e)
31        {
32           if ((Convert.ToInt32(comboBox1.Text) > 0) || (Convert.ToInt32(comboBox1.Text) < 255))
33                {
34                comboBox1.Items.Add(comboBox1.Text);
35                threshold = comboBox1.Text;
36                this.Close();
37                }
38                else
39                MessageBox.Show("Please Select a Threshold Limit For Binarization",
     "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
40        }
41     }
42   }
```

**Code Segment 2:** Binarization event present in Form1.cs.

method present in Binarization.cs is invoked and the GetValue() method returns the threshold binarization value (**Code Segment 1, Line 15-17**)

### Quality control

Some basic functional testing procedures were carried out for Image Enhancer 1.0.0 Release Candidate. The operating systems used for performance testing are Windows XP with SP3, Windows 7 professional (32 & 64 bit), Windows 7 ultimate (32 & 64 bit) and Windows 8/8.1 (32 & 64 bit). It was noticed that Image Enhancer performs equally well on these OS but the execution time of the software increases if each input image size exceeds 15 MB in size. Black box testing was also performed to ensure the proper functionality without peering into its internal structures or workings. In this context, each digital imaging filter was tested with different threshold values and the result data set was kept, which can be found here http://imageenhancer.codeplex.com/releases. Apart from these, Image Enhancer was also tested based on some crucial software engineering aspects (Code Metric) obtained directly from Microsoft Visual Studio 2010 Ultimate IDE, which are described below.

#### *Code Metric for Image Enhancer 1.0.0 Release Candidate*

**Maintainability Index (MI)** - An index value between 0 and 100 that represents the relative ease of maintaining the code. A high value means better maintainability. The classifications of the Maintainability index with respect to range of values are defined as

- 20-100-good maintainability
- 10-19-moderately maintained
- 0-9-low maintainability
- The MI value for Image Enhancer 1.0 RC was found to be 55. (**Ref. 1**)

**Cyclomatic/Conditional Complexity (CC)** - A software metric measuring linearly independent paths through a program's source code. The CC of this software was found to be 802. (**Ref. 1, 2**)

**Depth of Inheritance Tree (DIT)** – It is defined as the maximum length from the node to the root of the tree which was found to be 7 here. Since all classes inherit from System. Object class, the depth 7 means 6 child classes are utilized under System. Object. (**Ref. 1**)

**Class Coupling (Cc)** – It is a distinguishing metric and accurate predictor of software failure to depict how many classes a single class uses. It also measures the coupling to unique classes through parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields defined on external types, and attribute decoration. The class coupling value of Image Enhancer 1.0.0 RC was 191. (**Ref. 1**)

**Lines of Code (LC)** – 8309

The Release Candidate (RC) and the Beta versions of Image Enhancer went through an extensive acceptance testing. "SOFTPEDIA" and "FreewaresFiles" certified that Image Enhancer didn't contain any form of malware, including but not limited to: spywares, viruses, Trojans and backdoors. This software product was also thoroughly tested in SOFTPEDIA lab and was found absolutely clean. (Dated 16th July, 2014)

## (2) Availability

**Operating system**
Microsoft windows XP (SP3), Windows Vista, Windows 7, Windows 8/8.1

**Programming language**
Visual C# 4.0

**Additional system requirements**

- Memory: 512 MB RAM
- Disk Space: 3428 KB
- Processor: 1 GHZ
- Input Devices: USB/Surveillance Camera(If live video feed and snapshots are required)

**Dependencies**
Microsoft .Net Framework 4.0 or higher

**List of contributors**
Abhisek Hazra, Centre for Development of Advanced Computing, India, opensource@abhisek-hazra.in

Role: Conceptualization, Planning and Designing, Writing the software, Functional Testing, Software Packaging, Paper Writing.

**Archive**
***Name***
Figshare

***Persistent identifier***
http://dx.doi.org/10.6084/m9.figshare.1045339

***License***
GNU lesser general public license (LGPL) version 2.1

***Publisher***
Abhisek Hazra

***Date published***
04/06/2014(Release Candidate)

**Code Repository**
***Name***
CodePlex

***Identifier***
http://imageenhancer.codeplex.com

***License***
GNU Lesser General Public License (LGPL) version 2.1

***Date published***
28/05/2014(Release Candidate)

***Language***
English

## (3) Reuse potential

Being a standalone application, Image Enhancer 1.0.0 Release Candidate provides a wide potential and benefit for the researchers in the field of Image Processing, apart from a bigger set of general users who are interested in image manipulation tasks like noise removal, smoothing and sharpening of an image, edge preservation and enhancement, saturation and brightness correction, resizing an image etc. It is a useful tool to dive into several image processing algorithms as all the parameters of the digital imaging filters can be fully customized as per the choices of the end users. The following figure (**Fig. 4**) illustrates an example of editing the parameter values of saturation increment to obtain the desired result which can be effectively used to rectify low saturated noisy images. Another case (**Fig. 5**) is shown where the 'jittering radius' parameter of the jittering filter has been modified to obtain different effects of Jittering filter. An artistic effect can be applied to an input image by the use of Jittering filter which replaces each pixel of an image with a random pixel from an adjacent neighbourhood of the specified radius.

Currently Image Enhancer has an implementation of 64 simple and complex digital imaging filters for different problem instances which can be easily utilized to find research outcomes and preparing quality research publications. Some potential applications include image classification and soft biometric cues where the images need pre-processing to rectify noises and distortion. Face, ear, signature and other biometric applications require pre-processing of RGB images to obtain morphological/shape based feature extraction where it can be used as an offline utility to measure the threshold limits of the images. Sharpening, restoration (betterment of an image) and retrieval of images (seek for image of interest) in different applications/research also demands a lot of attention where Image Enhancer can play significant roles. Also the video buffering facility present in Image Enhancer can be utilized in surveillance and asset visibility applications. Users of this software can modify it accordingly as per their requirements, as all the classes are loosely coupled. This tool was developed for producing quality output images. Being free to distribute and copy, this tool

**Fig. 4**: Example of parameter customization, on the extreme left, the original image is shown, while the right ones reflect 25%, 50%, 75% increment in saturation respectively. © Playboy. All Rights Reserved.

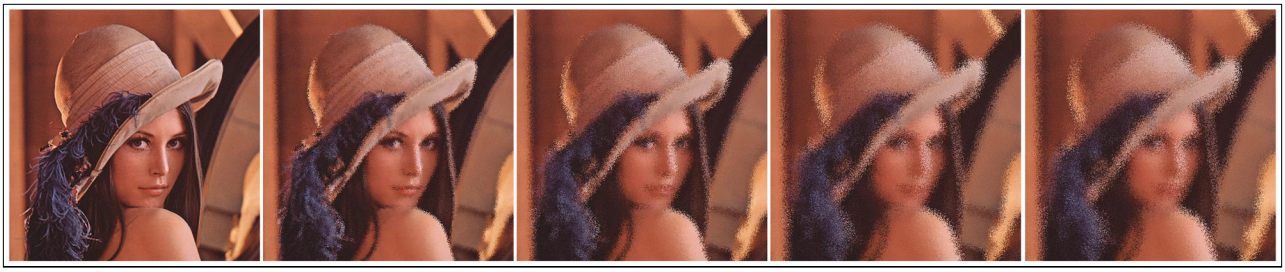

**Fig. 5**: Jittering parameter customization, on the extreme left, the original image is shown, while the right ones represent the images with 3, 6, 9 and 12 as their jittering radius in which pixels can move. © Playboy. All Rights Reserved.

could be of utter use and attraction to the undergraduate and graduate student community, aiming to start their research career in this field. Researchers working in computer vision and digital image processing in academia and industry will also find this software useful. Apart from this, it is also attractive for general end users interested in photography, multimedia, animation & graphic editing utility.

### Support Mechanisms for Image Enhancer

Image Enhancer is not part of any officially funded project, hence does not attain any donation or support from any agency or person. However, when developers, researchers or end users face problems related to code-reusing, compatibility and implementation, they are encouraged to post their issues in official project hosting page at CodePlex (http://imageenhancer.codeplex.com/discussions) or to contact the developer directly (Abhisek Hazra, opensource@abhisek-hazra.in) for urgent support.

### References


1. **Microsoft Developer Network** 2014 Code Metrics Values. Available at http://msdn.microsoft.com/en-us/library/bb385914.aspx.
2. **Wikipedia** 2014 Cyclomatic Complexity. Available at http://en.wikipedia.org/wiki/Cyclomatic_complexity.
3. **AForge.NET** 2014 AForge.NET. Available at http://www.aforgenet.com.
4. **Matalas, I, Benjamin, R** and **Kitney, K** 1997 An Edge Detection Technique Using the Facet Model and Parameterized Relaxation Labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 328-341.
5. **Smith, S M** and **Brady, J M** eds. 1997 SUSAN—A New Approach to Low Level Image Processing, 45-71.


**How to cite this article:** Abhisek, H 2014 Image Enhancer: A Graphic Editor to Apply Numerous Effects in Digital Image. *Journal of Open Research Software* 2:e31, DOI: http://dx.doi.org/10.5334/jors.bm

**Published:** 11 December 2014



]u[ *Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS